**LAB ASSIGNMENT 01**

**COMSATS University Islamabad**

Sahiwal Campus



**Usama Sarwar**

FA17-BS(CS)-090-B

**Mr. Ehtsham**

Compiler Construction

October 7, 2020

Table of Contents

[1. Task 1 1](#_Toc52922921)

[1.1 Code 1](#_Toc52922922)

[1.2 Output 2](#_Toc52922923)

[2. Task 02 3](#_Toc52922924)

[2.1 Code 3](#_Toc52922925)

[2.2 Output 5](#_Toc52922926)

# Task 1

## Code

import 'dart:io';

main() {

  var file = new File('./file.txt');

  // Create Text File

  file.create();

  // Write operation

  stdout.write('User Input: ');

  String \_input = stdin.readLineSync();

  file.writeAsStringSync(\_input + '\n', mode: FileMode.append);

  // Append Method

  stdout.write('User Input (Append): ');

  \_input = stdin.readLineSync();

  file.writeAsStringSync(\_input + '\n', mode: FileMode.append);

  // Read Operation

  stdout.write(file.readAsStringSync());

  // Find Word From File

  stdout.write('Search in File: ');

  \_input = stdin.readLineSync();

  if (file.readAsStringSync().contains(\_input)) {

    print('Found!');

  } else {

    print('Not Found!');

  }

  List \_characters = file.readAsStringSync().codeUnits.map((unit) {

    return String.fromCharCode(unit);

  }).toList();

  stdout.write('Alphabetic Characters: ');

  for (var i in \_characters) {

    if (RegExp('[a-z]', caseSensitive: false, multiLine: true).hasMatch(i)) {

      stdout.write(i + ' ');

    }

  }

  stdout.write('\nNumeric Characters: ');

  for (var i in \_characters) {

    if (RegExp('[0-9]', multiLine: true).hasMatch(i)) {

      stdout.write(i + ' ');

    }

  }

  stdout.write('\nSpecial Characters: ');

  for (var i in \_characters) {

    if (!RegExp('[a-z ' ' A-Z 0-9]').hasMatch(i)) {

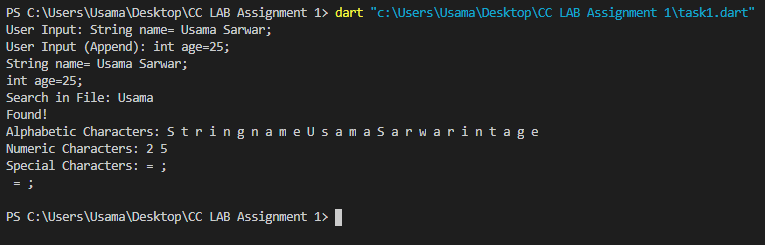
      stdout.write(i + ' ');

    }

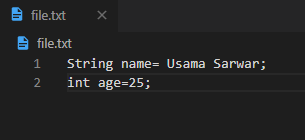
  }

}

## Output



Text File Created



# Task 02

## Code

#include <stdbool.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

// If the string is a VALID IDENTIFIER.

bool validIdentifier(char \*str)

{

    if (str[0] == '0' || str[0] == '1' || str[0] == '2' ||

        str[0] == '3' || str[0] == '4' || str[0] == '5' ||

        str[0] == '6' || str[0] == '7' || str[0] == '8' ||

        str[0] == '9' || isDelimiter(str[0]) == true)

        return (false);

    return (true);

}

// If the string is a KEYWORD.

bool isKeyword(char \*str)

{

    if (!strcmp(str, "if") || !strcmp(str, "else") || !strcmp(str, "while") || !strcmp(str, "do") || !strcmp(str, "break") || !strcmp(str, "continue") || !strcmp(str, "int") || !strcmp(str, "double") || !strcmp(str, "float") || !strcmp(str, "return") || !strcmp(str, "char") || !strcmp(str, "case") || !strcmp(str, "char") || !strcmp(str, "sizeof") || !strcmp(str, "long") || !strcmp(str, "short") || !strcmp(str, "typedef") || !strcmp(str, "switch") || !strcmp(str, "unsigned") || !strcmp(str, "void") || !strcmp(str, "static") || !strcmp(str, "struct") || !strcmp(str, "goto"))

        return (true);

    return (false);

}

// If the string is an INTEGER.

bool isInteger(char \*str)

{

    int i, len = strlen(str);

    if (len == 0)

        return (false);

    for (i = 0; i < len; i++)

    {

        if (str[i] != '0' && str[i] != '1' && str[i] != '2' && str[i] != '3' && str[i] != '4' && str[i] != '5' && str[i] != '6' && str[i] != '7' && str[i] != '8' && str[i] != '9' || (str[i] == '-' && i > 0))

            return (false);

    }

    return (true);

}

// If the character is a DELIMITER.

bool isDelimiter(char ch)

{

    if (ch == ' ' || ch == '+' || ch == '-' || ch == '\*' ||

        ch == '/' || ch == ',' || ch == ';' || ch == '>' ||

        ch == '<' || ch == '=' || ch == '(' || ch == ')' ||

        ch == '[' || ch == ']' || ch == '{' || ch == '}')

        return (true);

    return (false);

}

// If the character is an OPERATOR.

bool isOperator(char ch)

{

    if (ch == '+' || ch == '-' || ch == '\*' ||

        ch == '/' || ch == '>' || ch == '<' ||

        ch == '=')

        return (true);

    return (false);

}

// Extracts the SUBSTRING.

char \*subString(char \*str, int left, int right)

{

    int i;

    char \*subStr = (char \*)malloc(sizeof(char) \* (right - left + 2));

    for (i = left; i <= right; i++)

        subStr[i - left] = str[i];

    subStr[right - left + 1] = '\0';

    return (subStr);

}

// Parsing the input STRING.

void parse(char \*str)

{

    int left = 0, right = 0;

    int len = strlen(str);

    while (right <= len && left <= right)

    {

        if (isDelimiter(str[right]) == false)

            right++;

        if (isDelimiter(str[right]) == true && left == right)

        {

            if (isOperator(str[right]) == true)

                printf("'%c' IS AN OPERATOR\n", str[right]);

            right++;

            left = right;

        }

        else if (isDelimiter(str[right]) == true && left != right || (right == len && left != right))

        {

            char \*subStr = subString(str, left, right - 1);

            if (isKeyword(subStr) == true)

                printf("'%s' is a KEYWORD\n", subStr);

            else if (isInteger(subStr) == true)

                printf("'%s' is an INTEGER\n", subStr);

            else if (validIdentifier(subStr) == true && isDelimiter(str[right - 1]) == false)

                printf("'%s' is a valid IDENTIFIER\n", subStr);

            else if (validIdentifier(subStr) == false && isDelimiter(str[right - 1]) == false)

                printf("'%s' is not a valid IDENTIFIER\n", subStr);

            left = right;

        }

    }

    return;

}

// MAIN FUNCTION

int main()

{

    // maximum legth of program is 100 here

    char str[500] = "int age = 25 ;";

    parse(str);

    return (0);

}

## Output

